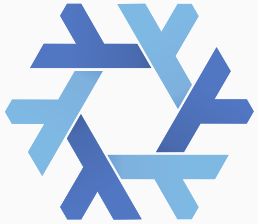


# NixOS

## Declarative Builds und Deployments

Hannes Schulze · 2024-05-04

## Was ist das?

- **Nix Expression Language:**
  Beschreibungssprache u.a. für Systemkonfigurationen
- **Nix (Package Manager):**
  Paketmanager auf Basis der Nix Expression Language
- **Nix Packages Collection/Nixpkgs:**
  Software-Repository für den Nix Package Manager
- **NixOS:**
  Linux-Distro, die den Nix Package Manager benutzt

## Was kann das?

- Ermöglicht es, das komplette System **deklarativ** zu konfigurieren
- Somit einfache Versionierung der Systemkonfiguration
- **Immutable:** Jede Änderung erzeugt neuen Snapshot
- **Reproduzierbar:** Pakete werden unabhängig von der Umgebung gebaut, teilweise auch gleiche Outputs
- Große Auswahl vorgefertigter Pakete in nixpkgs – auch für aarch64 :)

# Was kann das?

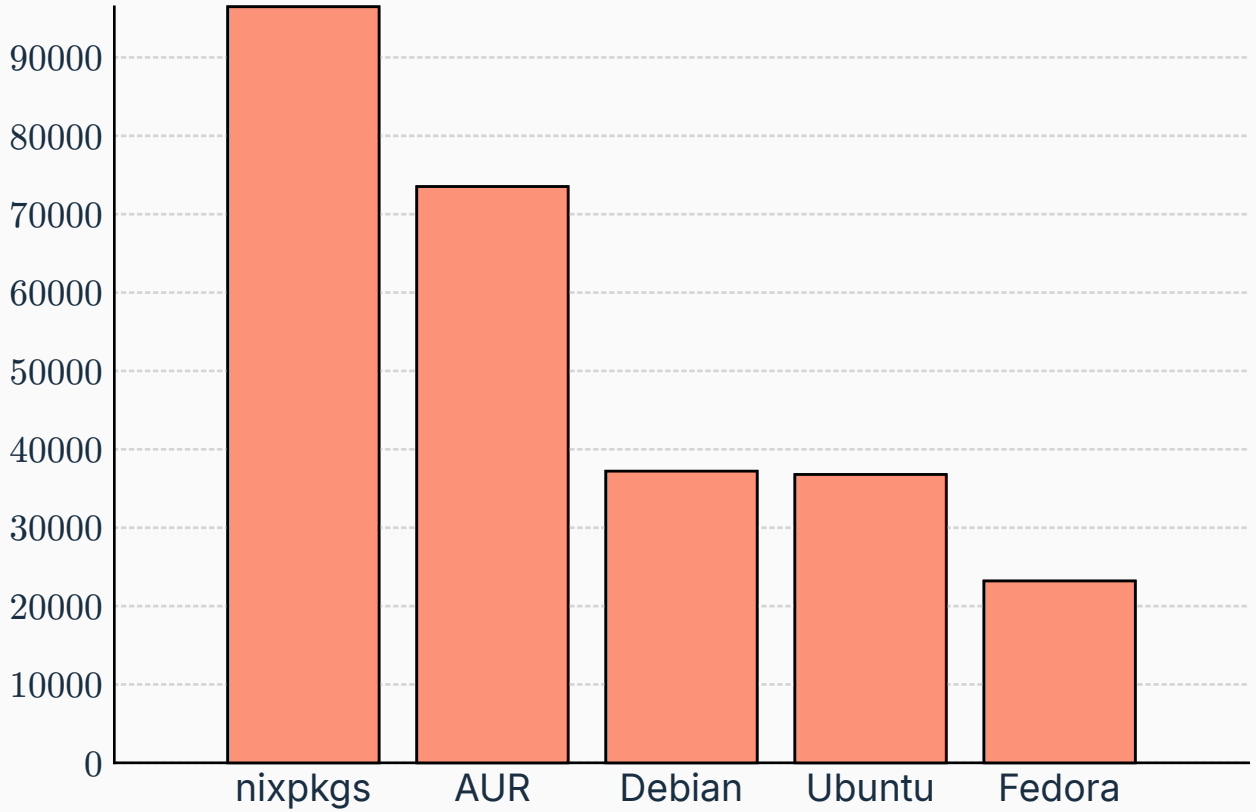



Figure 1: Anzahl von Paketen in verschiedenen Repositories
(jeweils aktuellste/unstable Version, Stand 2024-05-03, Daten von Repology)

# Was kann das?

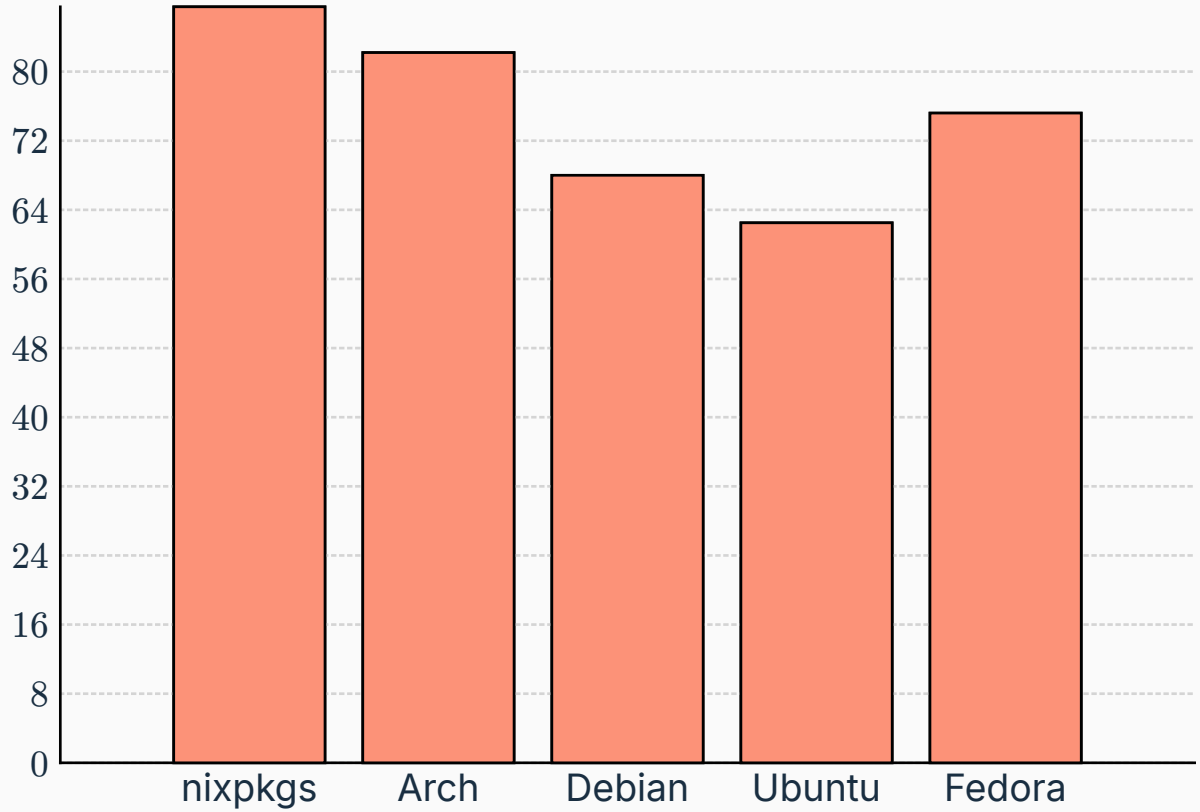



Figure 2: Prozentsatz aktueller Paketversionen in verschiedenen Repositories (jeweils aktuellste/unstable Version, Stand 2024-05-03, Daten von Repology)

## Für wen ist das?

- **Enthusiasten:**
  Anderer Ansatz als viele andere Systeme
- **Entwickler:**
  Development Shells vereinfachen Verwaltung von Abhängigkeiten
- **Administratoren:**
  Deklarative Konfiguration, einfaches Rollback

## Für wen ist das?

- Einmal konfiguriert, kann die Konfiguration immer wieder verwendet werden
- Allerdings zunächst starke Lernkurve

## Syntax

- Quelldateien enthalten genau einen **Ausdruck**
- Was dieser Ausdruck beschreibt und in welchem Format er ist, hängt von seinem Kontext ab
- Kann z.B. die komplette Konfiguration für ein System sein
- Die Sprache unterstützt unterschiedliche Ausdrücke

# Syntax

```
# Zeichenketten
"foo bar"
''
über mehrere
Zeilen verteilt
''
```

## Syntax

```
# Pfade
./pfad/zu/einer/datei
```

# Syntax

```
# Zahlen
42
1.69
```

## Syntax

```
# Listen
[ <expr> <expr> ... ]
[
  1
  2
  "foo"
]
```

# Syntax

```
# Attributmengen
{ key1 = <expr>; key2 = <expr>; ... }
{
  width = 20;
  height = 10;
}
```

## Syntax

- Vor dem eigentlichen Ausdruck können Variablen benannt und Berechnungen durchgeführt werden
- Das kann Schreibarbeit sparen
- Nix ist eine pure funktionale Programmiersprache

# Syntax

```
let
  format-event = year: "Linux-Tag ${toString year}";
  event = format-event 2024;
in
{
  name = event;
}

# Ausgewertet:

{ name = "Linux-Tag 2024"; }
```

# Pakete

- Pakete sind auch nur Ausdrücke
- Beschreibung, was getan werden muss, um das Paket zu bauen
- Fertig gebaute Pakete liegen im **Nix Store**
- Kennzeichnung mit Hash, mehrere Versionen desselben Pakets können wie bei Containern gleichzeitig installiert werden
- Nix Store ist world-readable (Vorsicht bei Secrets)

# Pakete

```
$ ls /nix/store
000ghm78048kh2prsfzkf93xm3803m0r-default.md
001av5zxcm99xpfrg2i477xil2c8x3q2-ed-1.20.1.drv
001gp43bjqzx60cg345n2slzg7131za8-nix-nss-open-files.patch
001wqi2dfw1jx2c9gpj5qfnwbbpxgllf-w3m-0.5.3+git20230121.drv
00844hz05lbdk3c4yhg7i23xd1sacx1m-nss-cacert-3.98-p11kit
00b0q85dlswb2jaz4lkdha8bmz67n0ra-thiserror-impl-1.0.51.drv
00gmdn2dvz6ykp8g2r8lgc63r2ji5w01-tasty-golden-2.3.5.drv
00q799gcyn2675cc2lfil3kh0i67w7fr-crate-thiserror-impl-1.0.51.tar.gz.drv
00qr10y7z2fcvrp9b2m46710nkjvj55z-update-autotools-gnu-config-scripts.sh
00r3wz6naq0nmrg7jnf2bgsppkf1ia6f-use-dynamic-system-antlr4-runtime.patch
00yysqmgvj3q7paym7022wdgrn5d9lak-home-manager-files
013hxba24zhdjf2nyqpjq3gfxsnwy3g8-libICE-1.1.1-dev
015i0y6q46v1s73a8j3k0mpl30bip4m4-keymap.drv
...
```

# Flakes

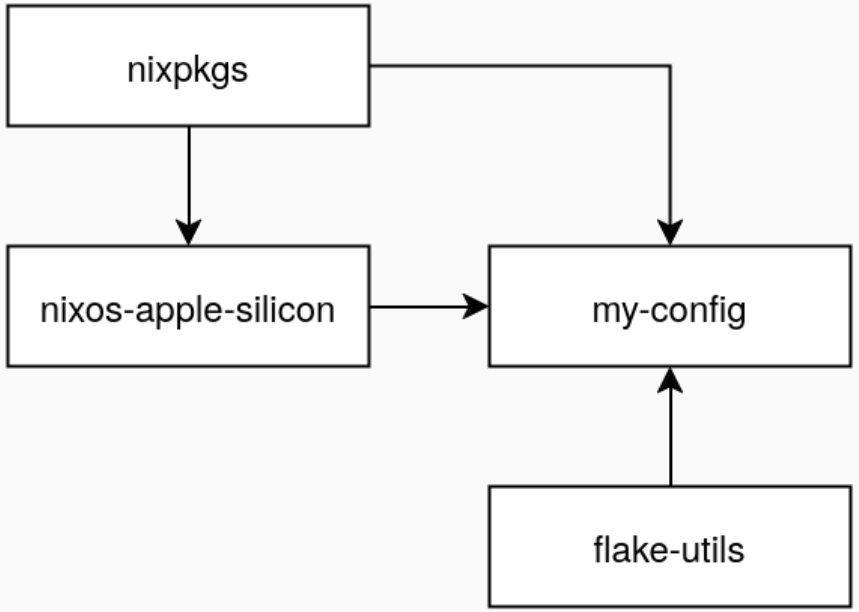
- Stellen als Ausgabe einen Ausdruck bereit
- Können andere Flakes als Eingaben haben
- Es gibt viele fertige Flakes aus unterschiedlichen Quellen
- Definition erfolgt in `flake.nix`, selbst ein Nix-Ausdruck

# Flakes

**Beispiele:**

- nixpkgs
- simple-nixos-mailserver
- Verschiedene Programme
- Die eigene Systemkonfiguration

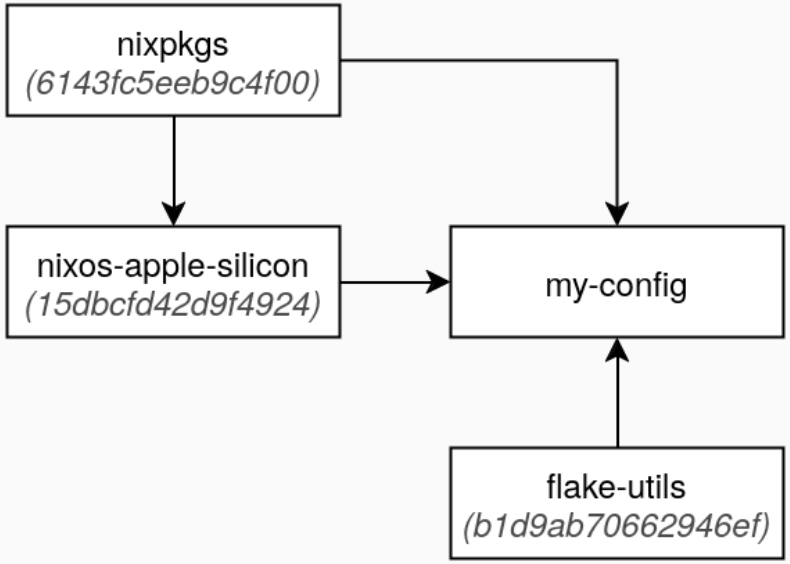
# Flakes



# Flakes

- Version von Abhängigkeiten ist fest, bei jedem Build gleich
- Ermöglichen so reproduzierbare Build-Umgebung
- Aktuelle Versionen stehen in `flake.lock`
- Aktualisieren mit `nix flake update`

# Flakes


nixpkgs
(6143fc5eeb9c4f00)
nixos-apple-silicon
(15dbcfd42d9f4924)
my-config
flake-utils
(b1d9ab70662946ef)

# Demo

- Installation mit `nixos-install`
- Aktualisieren der Konfiguration mit `nixos-rebuild`

# Ausblick

- Home-Manager
- Secret Management
- Modules